DESIGN OF MULTILEVEL NETWORKS

• TRANSFORMATIONS TO SATISFY CONSTRAINTS
  - number of gate inputs
  - network size
  - network delay

• DESIGN OF NETWORKS WITH XOR and XNOR GATES

• DESIGN OF NETWORKS WITH multiplexers (MUXes)
DESIGN MORE COMPLEX THAN FOR TWO-LEVEL NETWORKS

- NO STANDARD FORM

- SEVERAL REQUIREMENTS HAVE TO BE MET SIMULTANEOUSLY

- SEVERAL OUTPUTS HAVE TO BE CONSIDERED

- CAD TOOLS (logic synthesis) USED
A DESIGN PROCEDURE

1. OBTAIN SP or PS EXPRESSIONS FOR THE FUNCTIONS OF THE SYSTEM

2. TRANSFORM THE EXPRESSIONS (or the corresponding two-level networks) so that the requirements are met

3. REPLACE AND and OR GATES BY NAND and NOR WHEN APPROPRIATE

SEVERAL ITERATIONS MIGHT BE NEEDED
TYPICAL TRANSFORMATIONS TO MEET NETWORK REQUIREMENTS

• SIZE OF NETWORK: number of gates and number of gate inputs

• NUMBER OF GATES REDUCED BY

  1. FACTORING

  2. SUBEXPRESSIONS SHARED BY SEVERAL NETWORK OUTPUTS
EXAMPLE 6.1: 1-BIT COMPARATOR

INPUTS: \( x, y \in \{0, 1\} \)
\( c \in \{\text{GREATER, EQUAL, LESS}\} \)

OUTPUT: \( z \in \{\text{GREATER, EQUAL, LESS}\} \)

FUNCTION: 
\[
z = \begin{cases} 
\text{GREATER} & \text{if } x > y \text{ or } (x = y \text{ and } c = \text{GREATER}) \\
\text{EQUAL} & \text{if } x = y \text{ and } c = \text{EQUAL} \\
\text{LESS} & \text{if } x < y \text{ or } (x = y \text{ and } c = \text{LESS}) 
\end{cases}
\]
Example 6.1: Comparator (cont.)

Figure 6.1: COMPARATOR
Example 6.1: Comparator (cont.)

CODING:

<table>
<thead>
<tr>
<th>$c$</th>
<th>$c_2$</th>
<th>$c_1$</th>
<th>$c_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z$</td>
<td>$z_2$</td>
<td>$z_1$</td>
<td>$z_0$</td>
</tr>
</tbody>
</table>

- **GREATER**: 1 0 0
- **EQUAL**: 0 1 0
- **LESS**: 0 0 1

<table>
<thead>
<tr>
<th>$x, y$</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>100</td>
<td>100</td>
<td>001</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td>010</td>
<td>001</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td>001</td>
<td>001</td>
<td>100</td>
</tr>
</tbody>
</table>

$z$
Example 6.1: Comparator (cont.)

- SWITCHING EXPRESSIONS:

\[
\begin{align*}
    z_2 &= xy' + xc_2 + y'c_2 & G \\
    z_1 &= (x' + y)(x + y')c_1 & E \\
    z_0 &= x'y + x'c_0 + yc_0 & S
\end{align*}
\]

- RESULTING TWO-LEVEL NETWORK:
  - 7 AND and 4 OR gates
  - 22 equivalent gates
  - 25 gate inputs
REDUCING NETWORK SIZE (cont.)

DEFINE:

\[ t = (x + y') \]
\[ w = (x' + y) \]
\[ z_2 = xy' + tc_2 \]
\[ z_1 = twc_1 \]
\[ z_0 = x'y + wc_0 \]

- SIZE: 18 EQUIVALENT GATES

- FURTHER REDUCTION: NAND NETWORK – 9 EQUIVALENT GATES
Figure 6.2: 1-BIT COMPARATOR IMPLEMENTATIONS
EXAMPLE 6.2: MODULO-64 INCREMENTER

● A TWO-LEVEL IMPLEMENTATION:

\[ z_5 = x_5x'_4 + x_5x'_3 + x_5x'_2 + x_5x'_1 + x_5x'_0 + x'_5x_4x_3x_2x_1x_0 \]
\[ z_4 = x_4x'_3 + x_4x'_2 + x_4x'_1 + x_4x'_0 + x'_4x_3x_2x_1x_0 \]
\[ z_3 = x_3x'_2 + x_3x'_1 + x_3x'_0 + x'_3x_2x_1x_0 \]
\[ z_2 = x_2x'_1 + x_2x'_0 + x'_2x_1x_0 \]
\[ z_1 = x_1x'_0 + x'_1x_0 \]
\[ z_0 = x'_0 \]

● TWO-LEVEL NETWORK:

7 NOT, 20 AND, 5 OR gates, and 77 gate inputs
\[ z_5 = x_5'(x_4' + x_3' + x_2' + x_1' + x_0') + x_5'x_4x_3x_2x_1x_0 \]
\[ z_4 = x_4'(x_3' + x_2' + x_1' + x_0') + x_4'x_3x_2x_1x_0 \]
\[ z_3 = x_3'(x_2' + x_1' + x_0') + x_3'x_2x_1x_0 \]
\[ z_2 = x_2'(x_1' + x_0') + x_2'x_1x_0 \]
\[ z_1 = x_1x_0' + x_1'x_0 \]
\[ z_0 = x_0' \]

- **FOUR-LEVEL NETWORK (NOT-OR-AND-OR):**

7 **NOT** 10 **AND** and 9 **OR** gates, and 61 gate inputs
Figure 6.3: FOUR-LEVEL NETWORK FOR MODULO-64 INCREMENTER.
THE FAN-IN OF GATES

- FAN-IN OF GATES $\Leftrightarrow$ NUMBER OF OPERANDS PER OPERATOR

- REDUCED BY DECOMPOSING A LARGE GATE INTO SEVERAL SMALLER GATES

- AND AND OR ARE ASSOCIATIVE,

$$a + b + c + d + e + f = (a + b + c) + (d + e + f)$$
TERMS TO DECOMPOSE:

\[
(x'_4 + x'_3 + x'_2 + x'_1 + x'_0) = (x'_4 + x'_3 + r_{210})
\]
\[
(x'_5 x'_4 x'_3 x'_2 x'_1 x'_0) = x'_5 a_{43} a_{210}
\]
\[
(x'_3 + x'_2 + x'_1 + x'_0) = x'_3 + r_{210}
\]
\[
(x'_4 x'_3 x'_2 x'_1 x'_0) = x'_4 x'_3 a_{210}
\]

\[
\begin{align*}
    z_5 &= x_5 (x'_4 + x'_3 + r_{210}) + x'_5 a_{43} a_{210} \\
    z_4 &= x_4 (x'_3 + r_{210}) + x'_4 x'_3 a_{210} \\
    z_3 &= x_3 r_{210} + x'_3 a_{210} \\
    z_2 &= x_2 (x'_1 + x'_0) + x'_2 x'_1 x'_0 \\
    z_1 &= x_1 x'_0 + x'_1 x'_0 \\
    z_0 &= x'_0
\end{align*}
\]

- MORE GATES AND MORE LEVELS:
  
  6 NOT, 18 NAND, 3 NOR, size: 31 equivalent gates
Figure 6.4: REDUCING THE NUMBER OF GATE INPUTS
EXAMPLE 6.4: REDUCING OUTPUT LOAD OF A GATE (Buffering)

\[ z_i = w \times x \times y_i \quad 0 \leq i \leq 63 \]

Figure 6.5: REDUCING THE OUTPUT LOAD
• OUTPUT LOAD OF NAND PRODUCING $w \cdot x$: $64I$ ($I$ is load factor of NOR gate)

• PROPAGATION DELAY (high to low) between $x$ and $z_i$ (load 5 at output):

$$(0.05 + 0.038 \times 64) + (0.07 + 0.016 \times 5) = 2.63\text{ns}$$

• USE BUFFERS

<table>
<thead>
<tr>
<th>Gate type</th>
<th>Fan-in</th>
<th>Propagation delays</th>
<th>Input factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$t_{pLH}$ [ns]</td>
<td>$t_{pHL}$ [ns]</td>
</tr>
<tr>
<td>Buffer</td>
<td>1</td>
<td>$0.15 + 0.006L$</td>
<td>$0.19 + 0.003L$</td>
</tr>
<tr>
<td>Inv. Buf.</td>
<td>1</td>
<td>$0.04 + 0.006L$</td>
<td>$0.05 + 0.006L$</td>
</tr>
</tbody>
</table>

• DELAY:

$$(0.05 + 0.038 \times 4) + (0.15 + 0.006 \times 32) + (0.07 + 0.016 \times 5) = 0.69\text{ns}$$
EXAMPLE 6.5: EVEN PARITY CIRCUIT – alternatives

INPUT: \( x = (x_7, x_6, \ldots, x_0), \ x_i \in \{0, 1\} \)

OUTPUT: \( z \in \{0, 1\} \)

FUNCTION: \( z = \begin{cases} 
1 & \text{if } \sum_{i=0}^{7} x_i \text{ is even} \\
0 & \text{otherwise}
\end{cases} \)
IMPLEMENTATION 1: TWO-LEVEL NETWORK.

CSP: 128 MINTERMS – NO REDUCTION POSSIBLE

COST: 128 AND gates and one OR gate

EACH AND GATE 8 INPUTS, OR GATE 128 INPUTS

NOT PRACTICAL: large number of gates, large fan-in
IMPLEMENTATION 2: DIVIDE INTO TWO PARTS

\[ P(x) = P(x_l)P(x_r) + P'(x_l)P'(x_r) \]

Figure 6.6: NETWORK WITH FAN-IN=4
Table 6.2: CHARACTERISTICS OF ALTERNATIVE IMPLEMENTATIONS FOR THE PARITY FUNCTION

<table>
<thead>
<tr>
<th>Impl.</th>
<th>Network input load</th>
<th>Gates Type</th>
<th>Fan-in</th>
<th>Fan-out</th>
<th>Number</th>
<th>No. levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>64</td>
<td>AND</td>
<td>8</td>
<td>1</td>
<td>128</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OR</td>
<td>128</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>AND</td>
<td>4</td>
<td>1</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OR</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OR</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AND</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOT</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
EXAMPLE 6.6: 8-INPUT ODD-PARITY CHECKER

INPUT: \( x = (x_7, \ldots, x_0), x_i \in \{0, 1\} \)

OUTPUT: \( z \in \{0, 1\} \)

FUNCTION: \( z = \begin{cases} 0 & \text{if number of 1’s in } x \text{ is even} \\ 1 & \text{if number of 1’s in } x \text{ is odd} \end{cases} \)

\[ z = x_7 \oplus x_6 \oplus x_5 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus x_0 \]
EXAMPLE: 32-BIT EQUALITY COMPARATOR

INPUT: \( x = (x_{31}, \ldots, x_0) \), \( x_i \in \{0, 1\} \)
\( y = (y_{31}, \ldots, y_0) \), \( y_i \in \{0, 1\} \)

OUTPUT: \( z \in \{0, 1\} \)

FUNCTION: \( z = \begin{cases} 
1 & \text{if } x_i = y_i \text{ for } 0 \leq i \leq 31 \\
0 & \text{otherwise} 
\end{cases} \)

\[ z = AND(XNOR(x_{31}, y_{31}), \ldots, XNOR(x_i, y_i), \ldots, XNOR(x_0, y_0)) \]
Figure 6.8: 32-BIT EQUALITY COMPARATOR
• 2-INPUT multiplexer (MUX): $z = MUX[x_1, x_0, s] = x_1 \cdot s + x_0s'$

• SET $\{MUX\}$ IS UNIVERSAL (constants 0 and 1 available)

\[
\begin{align*}
NOT(x) &= MUX[0, 1, x] = 0 \cdot x + 1 \cdot x' = x' \\
AND(x_1, x_0) &= MUX[x_1, 0, x_0] = x_1x_0 + 0 \cdot x_0' = x_1x_0
\end{align*}
\]

Figure 6.9: 2-INPUT MULTIPLEXER AND NOT and AND GATES
SHANNON’S DECOMPOSITION (SD)

\[
f(x_{n-1}, x_{n-2}, \ldots, x_0) = f(x_{n-1}, x_{n-2}, \ldots, 1) \cdot x_0 \\
+ f(x_{n-1}, x_{n-2}, \ldots, 0) \cdot x'_0
\]

\[
z = f(x_{n-1}, x_{n-2}, \ldots, x_0) \\
= MUX[f(x_{n-1}, x_{n-2}, \ldots, x_1, 1), f(x_{n-1}, x_{n-2}, \ldots, x_1, 0), x_0]
\]

EXAMPLE:

\[
z = x_3(x_2 + x_0)x_1 = MUX[x_3x_1, x_3x_2x_1, x_0]
\]
**DESIGN OF NETWORKS WITH MUXes**

- **OBTAIN A TREE OF MULTIPLEXERS BY REPEATED USE OF SD**

![Diagram](image)

\[ z = f(x_{n-1}, \ldots, x_1, x_0) \]

*Figure 6.10: a) REALIZATION OF SHANNON’S DECOMPOSITION WITH MULTIPLEXER; b) REPEATED DECOMPOSITION.*
Example 6.8

- **IMPLEMENT** \( f(x_3, x_2, x_1, x_0) = z = x_3(x_1 + x_2x_0) \) WITH MUX TREE

- **DECOMPOSE WITH RESPECT TO** \( x_2, x_1, x_0 \)

\[
\begin{align*}
  f(x_3, 0, 0, 0) &= 0 &
  f(x_3, 0, 0, 1) &= 0 \\
  f(x_3, 0, 1, 0) &= x_3 &
  f(x_3, 0, 1, 1) &= x_3 \\
  f(x_3, 1, 0, 0) &= 0 &
  f(x_3, 1, 0, 1) &= x_3 \\
  f(x_3, 1, 1, 0) &= x_3 &
  f(x_3, 1, 1, 1) &= x_3
\end{align*}
\]

- **ELIMINATE REDUNDANT MUXes**
ORDERING OF VARIABLES IN SUBTREES AFFECTS THE NUMBER OF MUXes

Figure 6.11: